(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 398 640 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
23.04.1997 Bulletin 1997/17

(51) Int Cl.[6]: **G06F 17/30**

(21) Application number: **90305213.2**

(22) Date of filing: **15.05.1990**

(54) **Remote application interface**

Fernanwendungsschnittstelle

Interface d'application à distance

(84) Designated Contracting States:
DE FR GB

(30) Priority: **15.05.1989 US 352082**

(43) Date of publication of application:
22.11.1990 Bulletin 1990/47

(73) Proprietor: **International Business Machines Corporation**
Armonk, N.Y. 10504 (US)

(72) Inventors:
- **Devany, Estel Paul**
  Round Rock, Texas 78681 (US)
- **Garrison, Jane Ransom**
  Austin, Texas 78727 (US)
- **Jacobs, Dwayne Charles**
  Austin, Texas 78727 (US)
- **Jordan II, Lloyd Eugene**
  Austin, Texas 78729 (US)

(74) Representative: **Bailey, Geoffrey Alan**
IBM United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester Hampshire SO21 2JN (GB)

(56) References cited:
- **IBM SYSTEMS JOURNAL. vol. 27, no. 3, 1988, ARMONK, NEW YORK US pages 362 - 369 REINSCH 'Distributed database for SAA' Remote requests**
- **IEEE PROCEEDINGS OF THE FIFTH INTERNATIONAL CONFERENCE ON DATA ENGINEERING 10 February 1989, LOS ANGELES, USA pages 66 - 73 WOLSKI 'LINDA: A System for Loosely Integrated Databases' 5. NOTES ON IMPLEMENTATION**
- **MINI MICRO SYSTEMS vol. 11, no. 6, June 1988, BOSTON US pages 49 - 56 RAUCH- HINDIN 'LAN DBMS PIECES COME TOGETHER'**

EP 0 398 640 B1

Description

The present invention is related generally to digital computers, and more specifically to a system and method for executing application programs over a distributed network.

As small computers continue to become more powerful, and their costs decrease, networks of computers continue to become more common. These networks can be connected using a variety of network architectures, and typically consist of a moderate to large number of nodes. Each node can be a stand alone computer system, or a network shared resource such as a file server or printer.

In some networks, it is common for a user at one node to wish to execute a program or access data which resides on another node. Such execution or access can be accomplished in several different ways. The user can copy the necessary files from the remote node to his own local node, and process them locally. It is also possible to have the local node, typically a workstation or desktop computer, emulate a simple terminal, and access the remote node. Under the second arrangement, commands are entered from, and results displayed on, the local node, while all processing takes place on the remote node.

A third technique is to execute an applications program on the local node which communicates to the remote node in a manner transparent to the user. The local applications program can send commands to the remote node in order to access data or cause execution of programs on the remote node.

The techniques just described have several limitations and drawbacks. The technique of copying data and programs to a local node, not in general use on sophisticated networks, spends large amounts of time copying files which may be quite large in comparison to the amount of data actually needed. Also, creating multiple copies of files introduces a serious data coherency problem, in that it is difficult to reflect updates to a central location in a timely manner.

Using a local node to emulate a simple terminal minimizes the copying large files from one node to another, but still uses a fairly large share of network communication resources. Everything typed at the local terminal, and everything displayed thereon, requires transmission of information over the network. Using an applications program running on the local node to interface with a user and send encoded commands to the remote node can decrease the amount of information transmitted, but does not entirely eliminate the problem.

For example, it is common for a central database to be connected to a network for access by the other nodes. The database can be access with special commands, such as those used in a Structured Query Language (SQL). Each SQL statement defines a single request to the database. As used herein, a transaction is an integral piece of work which, when completed, is committed to the database. All changes to the database are tentative until committed, so that an integrated transaction can be rolled back, leaving the database in the same state it was before the transaction began. A series of database requests are generally needed to perform a single transaction.

IBM Systems Journal 27 (1988) No 3 pp 362 - 369 describes a system in which SQL database requests can be executed remotely by sending messages representing the requests over a communications network from an application running on a local machine to a data base management system on a remote machine.

When an application is running on a local node, and communicating with a database manager on a remote, or server node, each request in a transaction requires two communications over the network. The database request must first be transmitted from the local node to the database server, and the results must be returned to the local node. Thus, if a single transaction requires 7 database requests, 14 separate messages must be communicated over the communications network.

Viewed from a first aspect the invention provides a method for executing a transaction between a local network node and a remote node in a distributed database system, the local network node having a user interface facility and the remote node having a database manager for a remote database accessible through the remote node, the method comprising the steps of: enabling, via the user interface facility, a user to initiate a transaction; requiring, via the user interface facility, the user to enter items of information necessary for a series of requests defining the transaction; transmitting from the local network node over a network communications link to the remote node, as a single message, the information items specified by the user; responsive to receipt of the single message at the remote node, making a series of individual database requests through the database manager using the transmitted information items to generate a transaction result; and returning the transaction result to the local network node as a second single message over the communications link.

In preferred embodiments of the invention a system suitable for use on a computer network provides a user interface on a local node and an application to be run on a remote node. An application for accepting input from the user and translating it to appropriate commands for the remote application is divided, and located partially on the local node and partially on the remote node. That portion located on the local node gathers any information required from the user and transmits it to the portion located on the remote node in an efficient manner. The remote location portion uses the transmitted information to interface with the remote application and obtain results. The results are collected and transmitted to the local portion, from which they are returned to the user.

Viewed from a second aspect the invention provides a distributed database system in which a transac-

tion can be executed between a local network node and a remote node, the local network node having a user interface facility and the remote node having a database manager for a remote database accessible through the remote node, the local network node comprising: means for enabling, via the user interface facility, a user to initiate a transaction; means for requiring, via the user interface facility, the user to enter items of information necessary for a series of requests defining the transaction; means to transmit from the local network node over a network communications link to the remote node, as a single message, the information items specified by the user; the remote node comprising: means responsive to receipt of the single message for making a series of individual database requests through the database manager using the transmitted information items to generate a transaction result; and means for returning the transaction result to the local network node as a second single message over the communications link.

An embodiment of the invention will now be described, by way of example only, with reference to the accompanying drawings in which:

FIGURE 1 is a block diagram of a system according one embodiment of the present invention;

FIGURE 2 is a flowchart of a method for making database accesses in accordance with the system of **Figure 1**; and

FIGURES 3 and **4** illustrate data structures suitable for use with the method of **Figure 2**.

The preferred embodiment is described in terms of a system and method for remotely accessing a database over a network. As will be described below, the precise nature of the database and software for directly manipulating that database do not form a part of the present invention. However, the preferred embodiment will be described in relation to a database manager which accepts requests using a Structured Query Language (SQL) such as is available from International Business Machines Corporation.

Referring to **Figure 1**, a system for making remote database accesses includes a user interface **10**. The interface, typically including a display, keyboard, mouse or other pointing device, and software to drive these devices, is in communication with a user application (interface portion) **12**. The interface portion **12** includes software for accepting input from the user interface **10** and directing output thereto. Typically, a computer system on a network will have a single user interface **10**, with multiple user applications **12** which can be invoked by the user.

A remote data services software utility **14** can be invoked by the interface portion **12**, generally through a procedure call. The remote data services **14**, in turn, can invoke, via a procedure call, either a user application (server portion) **16** or a communications interface utility **18**. As is described below, the server portion **16** generates calls to a local database manager **20**, and accepts results returned therefrom. The database manager **20** accepts requests from the server portion **16** in a predetermined format, such as SQL requests, and performs reads and updates on a database. The details of the database and the database manager **20** do not form a part of the present invention. SQL database managers are commonly available, and many of these can be used with the present invention with little or no modification.

The communications interface utility **18** connects to a network represented by communications line **22**. A large number of other devices may be connected to the network as indicated by communication lines **24**, and one node in particular is connected through communications line **26** which is attached to a communications interface **28**. The type of network used does not form a part of the present invention, and the communications interfaces **18**, **28** are simply those which are appropriate to a given network environment. Many different commonly available network protocols are suitable for use with the present invention.

At the remote node, a remote data services software utility **30** communicates with communications interface **28**. The remote data services utility **30** also communicates with a user application (server portion) **32**, which in turn makes database requests to a database manager **34**. The communications between and operations of items **30**, **32**, and **34** is similar to that of items **14**, **16**, and **20**.

Figure 2 is a flowchart illustrating the sequence of events which occur when a user at the local node undertakes to perform a transaction on the database. As described above, a transaction is a sequence of individual database requests, with any changes to the database being committed only when the transaction has been successfully completed. Thus, the sequence of database requests making up a single transaction can be considered as a whole, with all the elements thereof completing successfully, or failing, together. Any updates made to the database do not actually take effect until the transaction commits.

Referring to **Figure 2**, when a user initiates a transaction, the interface portion of the user application gathers all of the required information from the user **40**. The interface portion of the application may require the user to enter several items of information in response to individual queries, the user may be required to fill in blanks on a template, or other techniques known in the art may be used. Once all of the information necessary for the transaction has been gathered, it is formatted into a standard format **42** as will be described below. At this time, the interface portion **12** makes a procedure call to the remote data services utility **14** and passes the formatted information thereto.

The services utility **14** first determines whether the database against which the transaction is to run is a lo-

cal or remote database **44.** If the database is not local, the services utility **14** prepares the data for transmission over a network **46.** This generally involves serializing what may be a complex data structure, including blocks of memory interconnected by pointers, into a "flat" structure representative of the same relationships. The data is then sent to the appropriate remote node **48** by the communications interfaces **18**, **28**, and the formatted data is recreated **50** at the remote node by the data services utility **30** at that node. The recreated data is preferably identical to the data formatted in step **42.**

The remote data services utility **30** then causes the server portion of the user application **32** to execute **52,** and passes the formatted data to it. The server portion of the user application now has all of the data necessary to execute the entire transaction. Until this time, the actual database requests which make up the transaction have not been considered by any part of the system. The code of the server portion **32** consists of a series of procedure calls to the database manager **34,** using the data gathered from the user as input. These procedure calls are database requests **54,** and control passes back and forth between the server portion of the user application **32** and the database manager **34.**

Once all of the database requests that make up a single transaction have been completed, the server portion of the user application **32** formats the results **56** and returns them to the services utility **30.** A check is made to see if the accessed database was located on the same local node as the user **58,** and if not the results are prepared for transmission **60** in the same manner as data was prepared for transmission in step **46**. The data is then sent to the user node **62**, and the formatted results are recreated **64** by the remote data services utility **14**. The results are then returned to the user application **66**, which performs local actions such as displaying the results to the user.

If the database to be accessed is a local database, the server portion **16** and database manager **20** are invoked on the local node rather than invoking the server portion **32** and database manager **34** on a remote node. The flow of control in **Figure 2** determined by steps **44** and **58** represents this situation. If the database is local to the user, the remote data services utility **14** invokes the server portion **16** directly, with no data preparation, transmission, or format recreation steps necessary. As far as the user interface **10** and interface portion **12** are concerned, the location of the server portion and database manager are not important; the information gathering and formatting steps **40**, **42** are the same in either case.

For a particular application, a database manager is invoked by only a single server portion of the user application. The server portion can be called by a user application interface portion running locally, or by any number of such interface portions running on different network nodes. The only difference between users running database transactions from a local node or remote nodes is that the remote data services utility **14** on the remote nodes cause data to be transmitted over the network instead of passed directly to the server portion **16**.

An example of the type of system which could advantageously be designed in accordance with the above principles would be a network of automated teller machines (ATM). A customer who wished to, for example, withdraw money from his account would initiate a transaction at an ATM by identifying himself with a magnetically coded card and a password. The card contains customer information such as bank identification and account number. The interface portion **12** requests the user to enter the amount of the transaction, and builds a data structure which generally includes at least the bank identification, account number, amount of transaction, and an identification of the ATM in use. This information is then transmitted to a central server holding the database. The server portion of the user application **32** then uses this transmitted information to make a series of calls to the database. Such a series of calls might include, for example, locking the required resources at the beginning of the transaction, updating the customer's account balance, updating the bank account balance, and updating the ATM account balance, and committing the transaction, and releasing the locked resources. A result is returned indicating whether the transaction is successful, and this information is transmitted back to the ATM. If the transaction is successful, the money is dispensed to the customer.

The example just described requires several calls to the database manager to perform various database functions. These include locking the necessary resources, performing the required updates, and committing the transaction. The program code to invoke these database requests is located in the server portion of the user application, so that the only information which need be transmitted over the network is the minimum amount of user information necessary for the transaction, and the results.

Figure 2 illustrates the sequence of events utilized to perform a single transaction. Establishing a network communications link between the user node and the remote node is not shown. This link can be established once for a series of transactions, can be established permanently, or may be established anew for each transaction. The technique chosen will depend on the nature of the network and its topology.

The preferred embodiment can also incorporate the features of the related European Published Patent Application No. entitled REMOTE INTERRUPT PROCESSING, a copy of which has been placed on the application file of the present invention. That application describes a technique for allowing the remote database manager to gracefully respond to an interrupt requested by the user. When a transaction is interrupted, preferably only the currently executing request is cancelled and rolled back, and the transaction remains pending. This means that all resource locks remain in place. The entire

transaction is cancelled and rolled back only upon receipt of an explicit command to do so after the above described interrupt.

In order to rollback only the current request, a savepoint, as known in the art, is taken as each new request is begun, as well as at the beginning of the entire transaction. Such partial rollback saves the time already invested in the completed requests if the transaction is restarted; only the time invested in a single request is lost.

Figure 3 shows a data structure of the type created by the user application interface portion **12** and utilized by the server portion **32**. **Figure 3** shows a structure for IN_SQLDA, which is an input data structure containing information needed for SQL database accesses. The variables shown in **Figure 3** are consistent with standard usage which will be recognized by those skilled in the art. The first two entries, SQLDAID and SQLDABC contain an identification string and total byte count for the structure. SQLN gives the number of variables which are included in the structure, and SQLD indicates how many of these are actually used. The entries SQLVAR [0] and SQLVAR[1] are pointers to data blocks containing information about variables. Each data block **70**, **72** corresponds to 1 variable, and identifies that variable in a manner consistent with standard SQL usage. For example, the type and length of the variable are shown, and a pointer to the actual data itself is contained in each data block **70**, **72**.

Figure 4 shows a data structure suitable for use for returning results as a variable OUT_SQLDA. This structure is analogous to that shown in **Figure 3**. Both IN_SQLDA and OUT_SQLDA can contain different numbers of variables from those shown in **Figures 3** and **4**, depending upon the requirements of the particular application.

When the database is located on a node remote from the user, the data structure shown in **Figures 3** and **4** must be "flattened" or "serialized" to a form suitable for transmission over a network. This serialization is performed by the remote data services routines **14**, **30**. The precise format used for the communication over a network will depend upon the type of network being used, but will generally be a simple serial string of characters. As long as all of the remote data services utilities know what communications format is being used, the precise nature of the transmission format is not important.

As will be recognized by those skilled in the art, the system and method described above minimize the amount of data which is transmitted over the network. The user application, which obtains data from the user and makes the necessary calls to the database, is divided into separate pieces in such a way as to allow for this minimum amount of communication. Obtaining user input, which can be time consuming given the relatively slow rate at which data is entered and the necessary validity checks which must be performed, is all accomplished at the local node without burdening the communications network. The process of performing database requests is all done at the server node at which the database is located. Use of the communications network is limited to identifying a transaction and passing precisely the information needed by that transaction, and returning a result.

It will be appreciated that in at least the described preferred embodiment the present invention provides for applications processing at a location remote from a user in such a manner as to minimize the amount of information communicated over a network so that only two messages need be communicated in order for multiple database access requests to be performed.

Claims

1. A method for executing a transaction between a local network node and a remote node in a distributed database system, the local network node having a user interface facility (10) and the remote node having a database manager (34) for a remote database accessible through the remote node, the method comprising the steps of:

   enabling, via the user interface facility (10), a user to initiate a transaction;

   requiring, via the user interface facility (10), the user to enter items of information necessary for a series of requests defining the transaction;

   transmitting (48) from the local network node over a network communications link (22, 26) to the remote node, as a single message, the information items specified by the user;

   responsive to receipt of the single message at the remote node, making (52, 54) a series of individual database requests through the database manager (34) using the transmitted information items to generate a transaction result;

   and returning (62) the transaction result to the local network node as a second single message over the communications link (22, 26).

2. A method as claimed in claim 1 wherein the information items and the transaction result are formatted (42, 56) in a preselected manner prior to transmission over the network communications link and wherein the information items and the transaction result are restored (50, 64) upon receipt at the remote node and at the local node, respectively.

3. A method as claimed in claim 1 or claim 2 wherein the database manager is an SQL database manager and each of the database requests comprises an

SQL request.

4. A distributed database system in which a transaction can be executed between a local network node and a remote node, the local network node having a user interface facility (10) and the remote node having a database manager (34) for a remote database accessible through the remote node, the local network node comprising:

means for enabling, via the user interface facility (10), a user to initiate a transaction;

means for requiring, via the user interface facility (10), the user to enter items of information necessary for a series of requests defining the transaction;

means to transmit (48) from the local network node over a network communications link (22, 26) to the remote node, as a single message, the information items specified by the user;

the remote node comprising:

means responsive to receipt of the single message for making (52, 54) a series of individual database requests through the database manager (34) using the transmitted information items to generate a transaction result;

and means for returning (62) the transaction result to the local network node as a second single message over the communications link (22, 26).

Patentansprüche

1. Verfahren zur Ausführung einer Transaktion zwischen einem lokalen Netzwerkknoten und einem entfernten Knoten in einem verteilten Datenbanksystem, wobei der lokale Netzwerkknoten eine Benutzerschnittstelleneinrichtung (10) besitzt und der entfernte Knoten einen Datenbankmanager (34) für eine entfernte Datenbank, auf die durch den entfernten Knoten zugegriffen werden kann, besitzt, wobei das Verfahren die folgenden Schritte umfaßt:

es über die Benutzerschnittstelleneinrichtung (10) einem Benutzer ermöglichen, eine Transaktion einzuleiten;

einen Benutzer über die Benutzerschnittstelleneinrichtung (10) auffordern, Datenfelder mit Informationen einzugeben, die für eine Reihe von Anfragen benötigt werden, die die Transaktion definieren;

Übertragen (48) der durch den Benutzer spezifizierten Informationsdatenfelder von dem lokalen Netzwerkknoten über eine Netzwerkkommunikationsverbindung (22, 26) zu dem entfernten Knoten als eine einzelne Nachricht;

Reagieren auf den Empfang der einzelnen Nachricht an dem entfernten Knoten, wobei eine Reihe individueller Datenbankanfragen über den Datenbankmanager (34) unter Nutzung der übertragenen Informationsdatenfelder durchgeführt werden (52, 54), um ein Transaktionsresultat zu erhalten;

und Zurückbringen (62) des Transaktionsresultats an den lokalen Netzwerkknoten als eine zweite einzelne Nachricht über die Kommunikationsverbindung (22, 26).

2. Verfahren, wie es in Anspruch 1 beansprucht wurde, wobei die Informationsdatenfelder und Transaktionsresultate vor der Übertragung über die Netzwerkkommunikationsverbindung auf eine vorher ausgewählte Weise formatiert werden (42, 56) und wobei die Informationsdatenfelder und die Transaktionsresultate bei Empfang am entfernten beziehungsweise lokalen Knoten rückgespeichert werden (50, 64).

3. Verfahren, wie es in Anspruch 1 oder Anspruch 2 beansprucht wird, wobei der Datenbankmanager ein SQL-Datenbankmanager ist und jede der Datenbankanfragen eine SQL-Anfrage enthält.

4. Verteiltes Datenbanksystem, bei dem eine Transaktion zwischen einem lokalen Netzwerkknoten und einem entfernten Knoten ausgeführt werden kann, wobei der lokale Netzwerkknoten eine Benutzerschnittstelleneinrichtung (10) und der entfernte Knoten einen Datenbankmanager (34) für eine entfernte Datenbank besitzt, auf die über den entfernten Knoten zugegriffen werden kann, wobei der lokale Netzwerkknoten folgendes umfaßt:

Mittel, um es einem Nutzer über die Benutzerschnittstelleneinrichtung (10) zu ermöglichen, eine Transaktion einzuleiten;

Mittel, um einen Benutzer über die Benutzerschnittstelleneinrichtung (10) aufzufordern, Datenfelder mit Informationen einzugeben, die für eine Reihe von Anfragen benötigt werden, die die Transaktion definieren;

Mittel zum Übertragen (48) der durch den Benutzer spezifizierten Informationsdatenfelder von dem lokalen Netzwerkknoten über eine Netzwerkkomminikationsverbindung (22, 26)

zu dem entfernten Knoten als eine einzelne Nachricht.

sowie der entfernte Knoten folgendes umfaßt:

Mittel zum Reagieren auf den Empfang der einzelnen Nachricht zur Erstellung (52, 54) einer Reihe individueller Datenbankanfragen über den Datenbankmanager (34) unter Nutzung der übertragenen Informationsdatenfelder, um ein Transaktionsresultat zu erzeugen;

und Mittel zum Zurückbringen (62) des Transaktionsresultats an den lokalen Netzwerkknoten als eine zweite einzelne Nachricht über die Kommunikationsverbindung (22, 26).

Revendications

1. Un procédé d'exécution d'une transaction entre un noeud de réseau local et un noeud distant dans un système de base de données distribué, le noeud de réseau local ayant une ressource d'interface utilisateur (30) et le noeud distant ayant un gestionnaire de base de données (34) destiné à une base de données distante accessible par l'intermédiaire d'une distance, le procédé comprend les étapes consistant à :

   permettre, via la ressource d'interface utilisateur (30), à un utilisateur d'initier une transaction;
   demander, via la ressource d'interface utilisateur (30), à l'utilisateur d'introduire des éléments d'information nécessaires pour une série de requêtes définissant la transaction;
   transmettre (48) depuis le noeud de réseau local sur une liaison de communication de réseau (22, 26) vers le noeud distant, à titre de message unique, les éléments d'information spécifiés par l'utilisateur;
   en réponse à la réception du message unique au noeud distant, établir (52, 54) une série de requêtes de base de données individuelles, par l'intermédiaire du gestionnaire de base de données (34), avec utilisation des éléments d'information transmis pur générer un résultat de transaction;

   et retourner (62) le résultat de transaction au noeud de réseau local à titre de deuxième message unique sur la liaison de communication (22, 26).

2. Un procédé selon la revendication 1, dans lequel les éléments d'information et le résultat de transaction sont formatés (42, 56) d'une manière présélectionnée avant la transmission sur la liaison de communication de réseau, et dans lequel les éléments d'information et le résultat de transaction sont rétablis (50, 64) à la réception au noeud distant et au noeud local, respectivement.

3. Un procédé selon la revendication 1 ou 2, dans lequel le gestionnaire de base de données est un gestionnaire de base de données SQL et chacune des requêtes de base de données comprend une requête SQL.

4. Un système de base de données distribué, dans lequel une transaction peut être exécutée entre un noeud de réseau local et un noeud distant, le noeud de réseau local ayant une ressource d'interface utilisateur (10) et le noeud distant ayant un gestionnaire de base de données (34) pour une base de données distante accessible par l'intermédiaire du noeud distant, le noeud de réseau local comprenant :

   des moyens pour permettre, via la ressource d'interface utilisateur (10), à un utilisateur d'initier une transaction;
   des moyens pour faire requête, via la ressource d'interface utilisateur (10) par l'utilisateur d'introduire des éléments d'information nécessaire pour une série de requêtes définissant la transaction;
   des moyens de transmission (48), depuis le noeud de réseau local, sur une liaison de communication de réseau (22, 26), au noeud distant, à titre de message unique, des éléments d'information spécifiés par l'utilisateur;

   le noeud distant comprenant :

   des moyens réagissant à la réception du message unique pour établir (52, 54) une série de requêtes de base de données individuelles par l'intermédiaire du gestionnaire de base de données (34) faisant utilisation des éléments d'information transmis pour générer un résultat de transaction;
   et des moyens de retour (62) du résultat de transaction au noeud de réseau local, à titre de deuxième message unique sur la ligne de communication (22, 26).

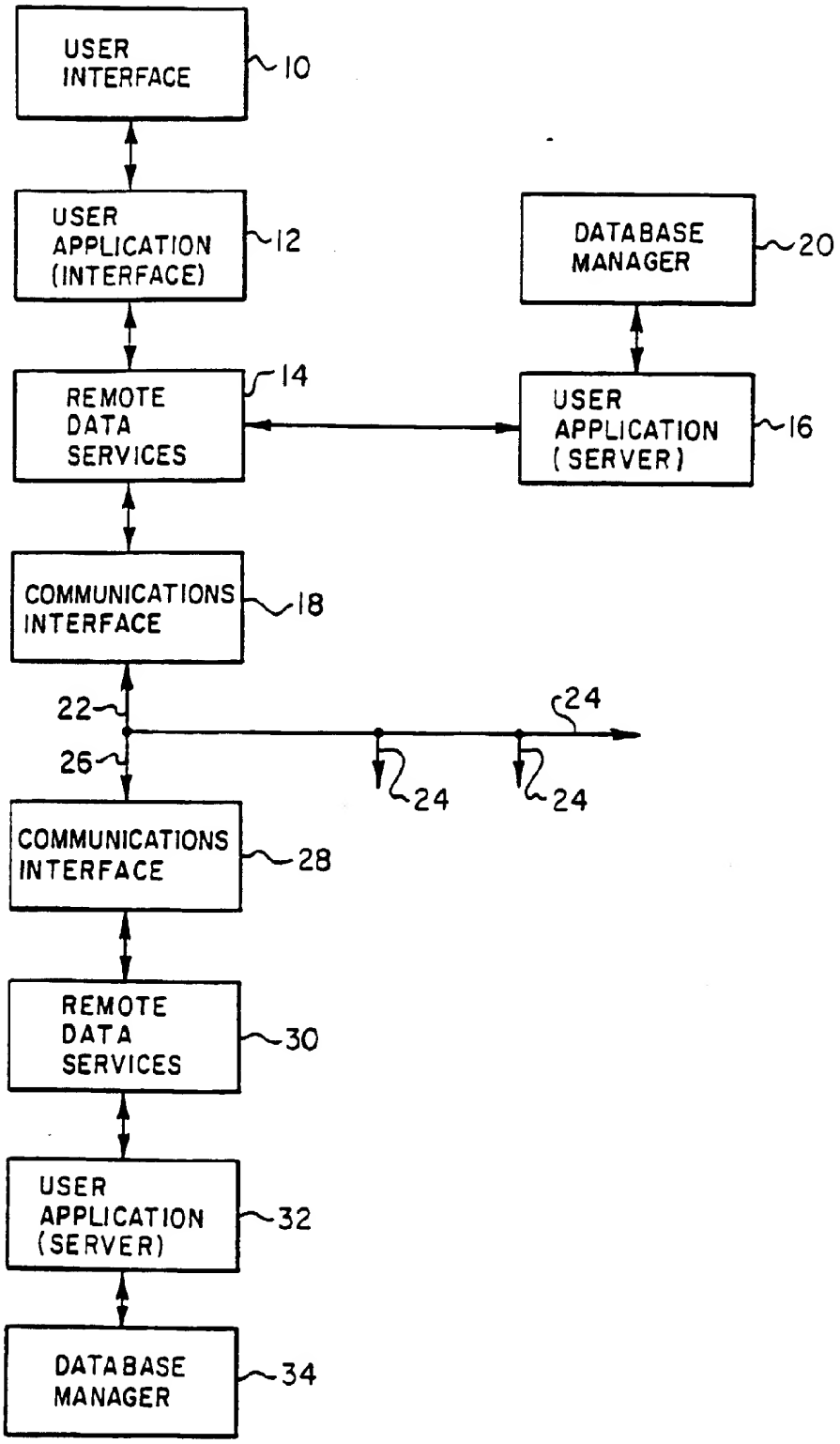
USER INTERFACE
10
USER APPLICATION (INTERFACE)
12
DATABASE MANAGER
20
REMOTE DATA SERVICES
14
USER APPLICATION (SERVER)
16
COMMUNICATIONS INTERFACE
18
22
24
26
24
24
COMMUNICATIONS INTERFACE
28
REMOTE DATA SERVICES
30
USER APPLICATION (SERVER)
32
DATABASE MANAGER
34

Fig. 1

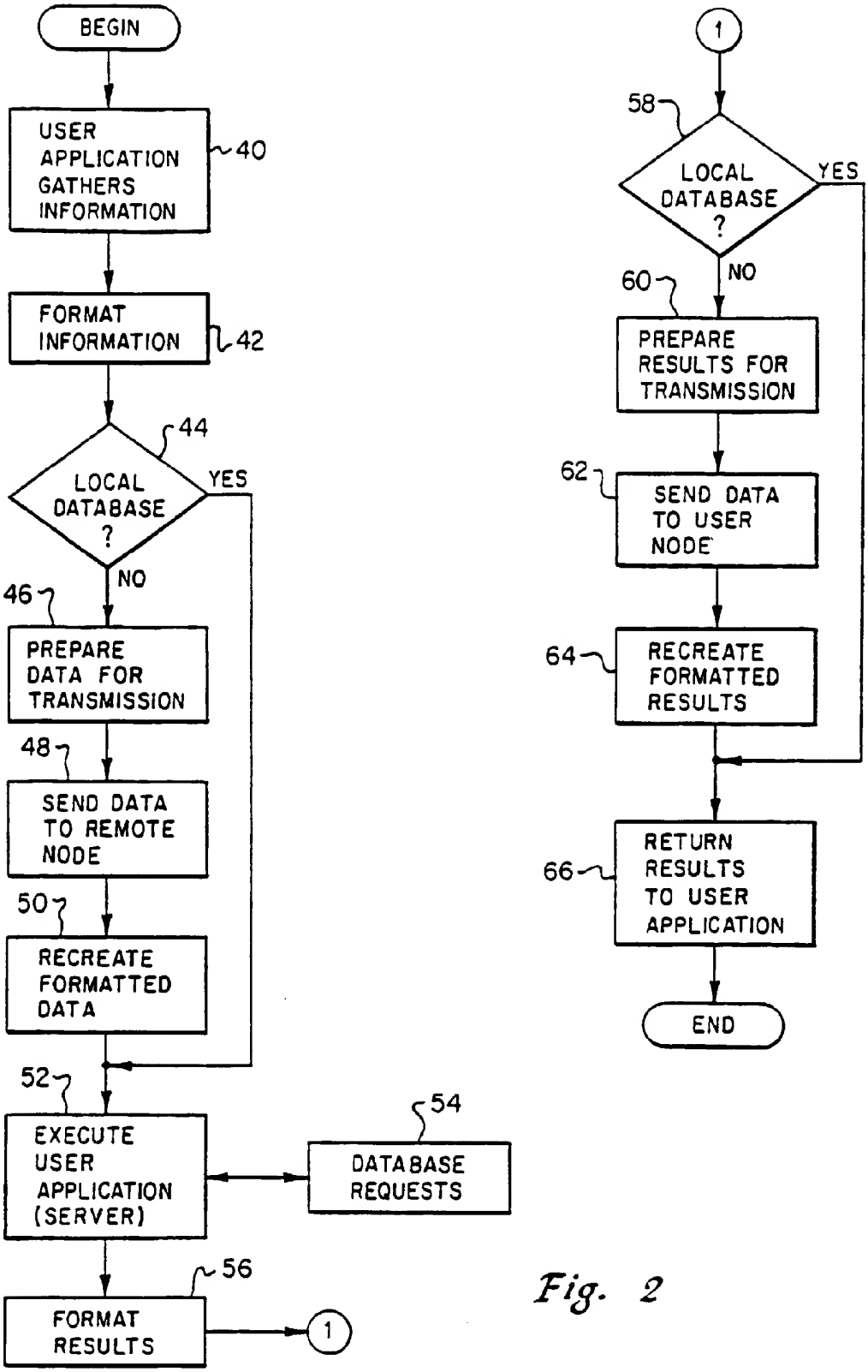
BEGIN
USER APPLICATION GATHERS INFORMATION
40
FORMAT INFORMATION
42
44
LOCAL DATABASE ?
YES
NO
46
PREPARE DATA FOR TRANSMISSION
48
SEND DATA TO REMOTE NODE
50
RECREATE FORMATTED DATA
52
EXECUTE USER APPLICATION (SERVER)
54
DATABASE REQUESTS
56
FORMAT RESULTS
1
1
58
LOCAL DATABASE ?
YES
NO
60
PREPARE RESULTS FOR TRANSMISSION
62
SEND DATA TO USER NODE
64
RECREATE FORMATTED RESULTS
66
RETURN RESULTS TO USER APPLICATION
END

Fig. 2

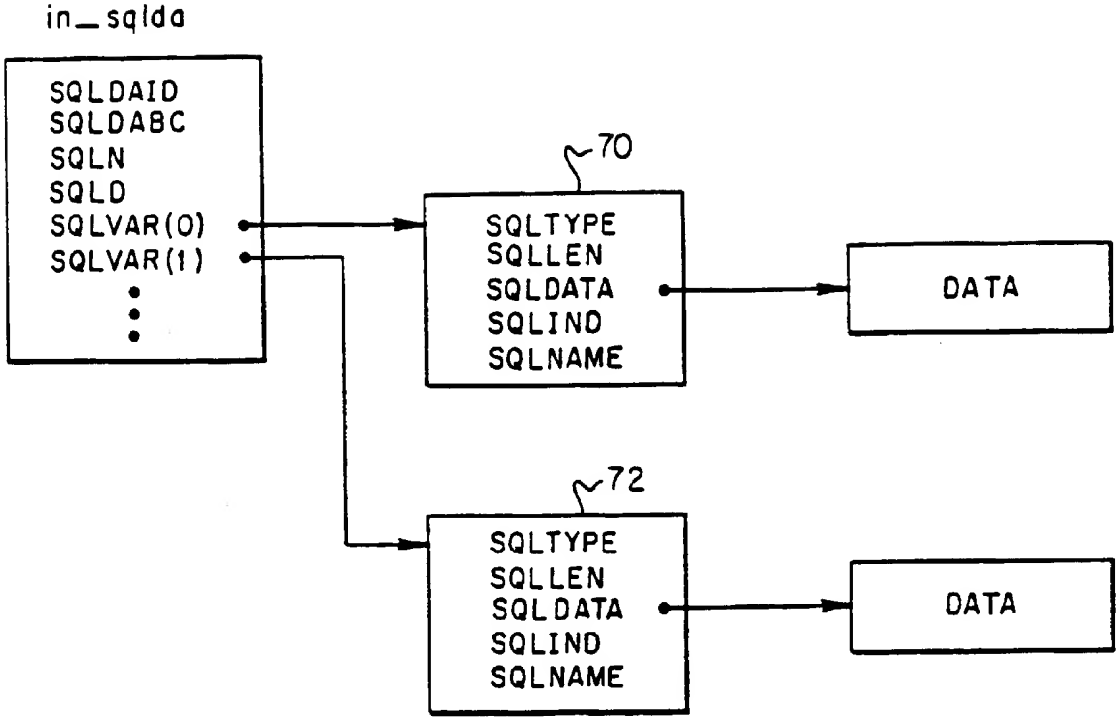
in_sqlda
SQLDAID
SQLDABC
SQLN
SQLD
SQLVAR(0)
SQLVAR(1)
70
SQLTYPE
SQLLEN
SQLDATA
SQLIND
SQLNAME
DATA
72
SQLTYPE
SQLLEN
SQLDATA
SQLIND
SQLNAME
DATA

Fig. 3

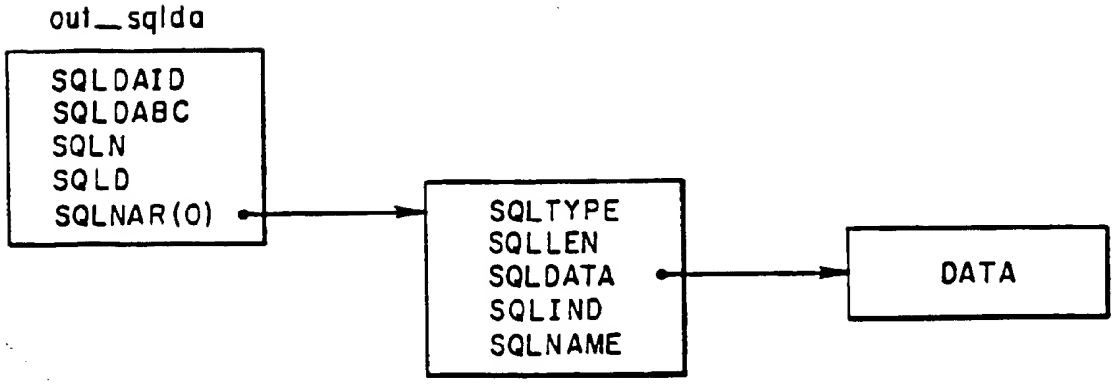
out_sqlda
SQLDAID
SQLDABC
SQLN
SQLD
SQLNAR(0)
SQLTYPE
SQLLEN
SQLDATA
SQLIND
SQLNAME
DATA

Fig. 4